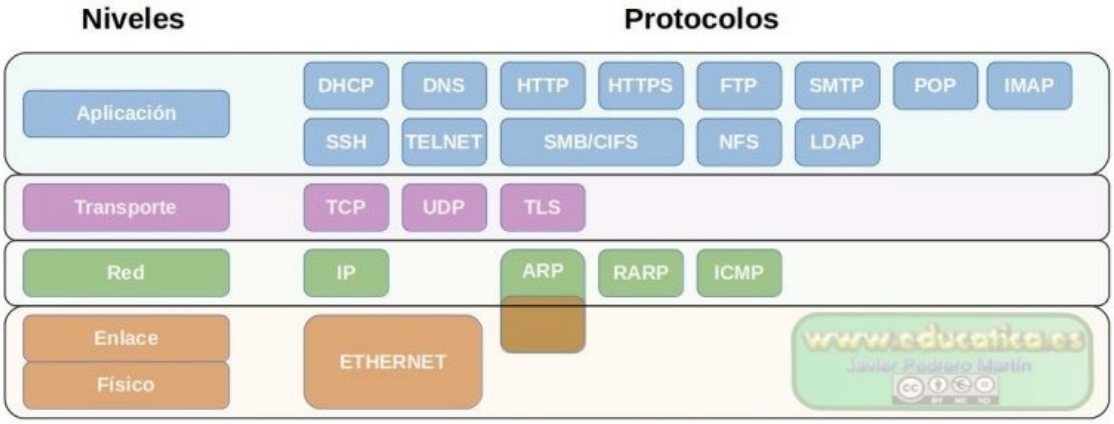
ÍNDICE

EJERCICIOS

1. Protocolos de comunicaciones: IP, TCP, HTTP, HTTPS.

Estos cuatro protocolos son pilares del modelo TCP/IP y de Internet. IP (Capa de Internet) se encarga de asignar direcciones y enrutar paquetes. TCP (Capa de Transporte) asegura que los paquetes lleguen completos y en orden. HTTP (Capa de Aplicación) permite la transferencia de hipertexto (páginas web). HTTPS es la extensión segura de HTTP que añade una capa de cifrado (SSL/TLS) para proteger la privacidad de los datos.



Los protocolos de comunicaciones son las normas y procedimientos formales que definen cómo los dispositivos de una red deben formatear, transmitir y recibir datos. Operan en capas, donde los protocolos de nivel superior dependen de los protocolos de nivel inferior para funcionar. Los protocolos IP, TCP, HTTP y HTTPS representan las capas clave de este sistema, conocido comúnmente como la pila de protocolos TCP/IP.

Protocolo de Internet (IP)

El IP (Internet Protocol) es el protocolo fundamental de la capa de red (o Internet en el modelo TCP/IP). Su función principal es el direccionamiento y el enrutamiento. Cada dispositivo conectado a Internet debe tener una dirección IP única (ej: 192.168.1.1 en IPv4 o una cadena más larga en IPv6) que funciona como su dirección postal. IP toma los paquetes de datos y determina la mejor ruta a través de la red (Internet) para que lleguen a esa dirección de destino, sin garantizar su llegada ni su orden.

Protocolo de Control de Transmisión (TCP)

El TCP (Transmission Control Protocol) opera en la capa de transporte y es el socio de IP. A diferencia de IP, TCP es un protocolo orientado a la conexión y fiable. Esto significa que:
- Establece una conexión entre el origen y el destino (el famoso three-way handshake).
- Divide los datos de la aplicación en segmentos.
- Asegura la entrega mediante el uso de acuses de recibo (ACK). Si un paquete se pierde, TCP solicita su reenvío.
- Reensambla los paquetes en el orden correcto en el destino.

TCP es esencial para aplicaciones donde la integridad de los datos es crítica, como la navegación web y la transferencia de archivos.

Protocolo de Transferencia de Hipertexto (HTTP)

El HTTP (Hypertext Transfer Protocol) es un protocolo de la capa de aplicación. Es la

columna vertebral de la World Wide Web. Define cómo se formatean y transmiten los mensajes, y qué acciones deben tomar los servidores web y los navegadores en respuesta a varios comandos. HTTP funciona bajo un modelo de solicitud/respuesta: el cliente (navegador) envía una solicitud (ej: GET para una página) y el servidor envía una respuesta (que incluye un código de estado, ej: 200 OK, y el cuerpo de la página web). Tradicionalmente, HTTP opera sobre TCP en el puerto 80 y transmite la información en texto plano, sin cifrar.

Protocolo Seguro de Transferencia de Hipertexto (HTTPS)

El HTTPS (Hypertext Transfer Protocol Secure) no es un protocolo totalmente distinto, sino el uso de HTTP sobre una capa de seguridad. Opera sobre TCP en el puerto 443. La diferencia clave es que integra los protocolos SSL (Secure Sockets Layer) o su sucesor, TLS (Transport Layer Security).

HTTPS establece una conexión segura y cifrada entre el cliente y el servidor. Esto asegura la confidencialidad (nadie puede leer los datos interceptados), la integridad (los datos no se modificaron en tránsito) y la autenticación (el usuario sabe que está hablando con el servidor correcto mediante un certificado de seguridad). Hoy en día, HTTPS es un estándar de facto para cualquier sitio web.

Tutorial (Conceptos básicos de redes):
https://developer.mozilla.org/es/docs/Learn/Common_questions/Como_funciona_Internet
(Mozilla Developer Network - Visión general de Internet)

Cloudflare. (s.f.). ¿Qué es HTTP? Recuperado de URL:
https://www.cloudflare.com/es-es/learning/ddos/glossary/hypertext-transfer-protocol-http/

IBM. (s.f.). Conceptos acerca de Internet, TCP/IP y HTTP. Recuperado de URL:
https://www.ibm.com/docs/es/cics-ts/6.x?topic=support-internet-tcpip-http-concepts

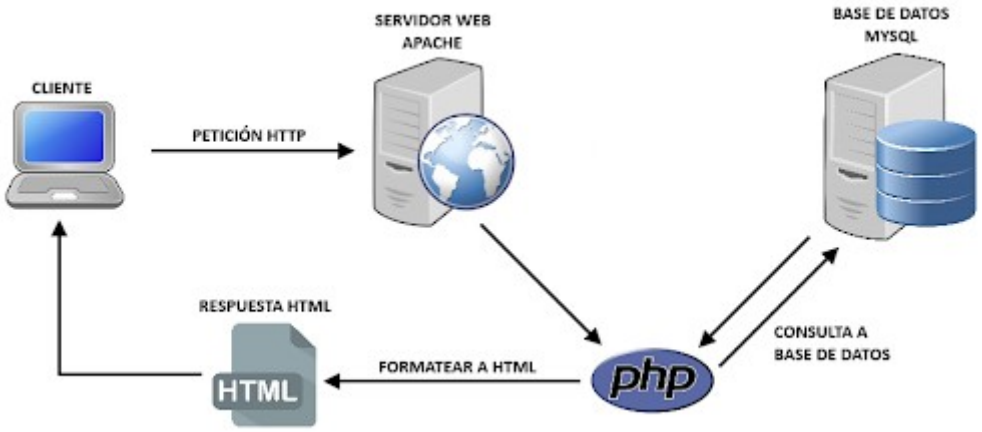
AWS. (s.f.). HTTP y HTTPS: diferencia entre los protocolos de transferencia. Recuperado de URL: https://aws.amazon.com/es/compare/the-difference-between-https-and-http/

2. Modelo de comunicaciones cliente – servidor y su relación con las aplicaciones web.

El modelo cliente-servidor es una arquitectura de red donde un cliente (solicitante) pide recursos o servicios, y un servidor (proveedor) responde a esas peticiones, estableciendo una división de tareas.

El Cliente (Solicitante): Es el componente que inicia la comunicación enviando una solicitud de servicio. Por lo general, ofrece la interfaz de usuario. En el contexto de Internet, el cliente más común es el navegador web (Chrome, Firefox, Safari), una aplicación móvil o cualquier otro software que necesite acceder a un recurso remoto.

El Servidor (Proveedor): Es el componente que espera pasivamente las solicitudes de múltiples clientes. Su función es procesar la solicitud, acceder a los recursos necesarios (como bases de datos o archivos), ejecutar la lógica de negocio y devolver una respuesta. Los servidores suelen ser equipos potentes con recursos centralizados.



Funcionamiento de la Comunicación
La interacción siempre sigue un ciclo de Solicitud-Respuesta:

Solicitud: El cliente formula una petición (ej. el usuario hace clic en un enlace) utilizando un protocolo de red (como HTTP, FTP, o SMTP).

Procesamiento: El servidor recibe la solicitud, la analiza, ejecuta la acción requerida (ej. consulta una base de datos) y genera una respuesta.

Respuesta: El servidor envía la respuesta de vuelta al cliente. En una aplicación web, esta respuesta suele ser código HTML, CSS, JavaScript o datos estructurados (como JSON).

Renderización: El cliente (navegador) recibe la respuesta y la interpreta para presentar la información al usuario.

Relación con las Aplicaciones Web
La aplicación web es la manifestación más extendida y visible del modelo Cliente-Servidor.

Cliente (Navegador): Se encarga de la Capa de Presentación (el Frontend). Es donde reside el código HTML, CSS y JavaScript que se encarga de la interfaz de usuario, la interactividad local y la formulación de las peticiones.

Servidor (Web Server): Se encarga de la Lógica de Negocio y la Capa de Datos (el Backend). Gestiona el almacenamiento de datos, las cuentas de usuario, el procesamiento de transacciones y la generación dinámica de contenido que se enviará al cliente.

El uso del protocolo HTTP/HTTPS es crucial: cuando un usuario introduce una URL, el navegador (cliente) envía una solicitud HTTP al servidor web, que responde con la página web o recurso solicitado, cerrando el ciclo cliente-servidor para esa interacción. Esto permite la centralización de la gestión de recursos (bases de datos) y la escalabilidad, ya que se pueden añadir más servidores para manejar más clientes.

Video (Explicación Sencilla): Modelo Cliente - Servidor ¿Qué es? Explicación Sencilla – YouTube

https://www.youtube.com/watch?v=KA7Ngcgth0Q

Visión General Cliente-Servidor
https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/First_steps/Client-Server_overview

Arsys Blog - Arquitectura cliente-servidor https://www.arsys.es/blog/todo-sobre-la-arquitectura-cliente-servidor

Kinsta - Arquitectura de Aplicaciones Web https://kinsta.com/es/blog/arquitectura-aplicaciones-web/

IONOS. (2023). ¿Cómo funciona el modelo cliente-servidor? Recuperado de URL: https://www.ionos.com/es-us/digitalguide/servidores/know-how/modelo-cliente-servidor/

3. Estudio sobre los métodos de petición HTTP /HTTPS más utilizados.

Los métodos HTTP/HTTPS más utilizados son **GET** (para solicitar datos) y **POST** (para enviar datos, como formularios o crear recursos).

Los métodos de petición, también conocidos como verbos HTTP, son la parte fundamental de una solicitud entre un cliente (como un navegador) y un servidor web. Indican la acción que el cliente desea realizar sobre un recurso identificado por una URL.

Aunque existen varios métodos (como CONNECT, OPTIONS, TRACE), los más utilizados en la web para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) son:

GET (Recuperar - Read):

- Función: Solicita una representación del recurso especificado. Se utiliza exclusivamente para recuperar datos.
- Características: Es un método seguro (no altera el estado del servidor) e idempotente (repetirlo no tiene efectos secundarios distintos al de la primera vez). Los parámetros se envían en la URL (Query String). Es cacheable.
- Ejemplo: Navegar a una página web (/index.html) o buscar un producto específico.

POST (Crear - Create):

- Función: Se utiliza para enviar datos a un recurso específico, a menudo para crear un nuevo recurso o realizar una acción que altere el estado del servidor.
- Características: No es seguro (modifica el estado del servidor) ni idempotente (repetir la petición podría crear duplicados). Los datos se envían en el cuerpo (body) de la solicitud, no en la URL. No es cacheable (excepto condicionalmente).
- Ejemplo: Enviar un formulario de registro o publicar un comentario.

PUT (Actualizar - Update/Reemplazar):

- Función: Reemplaza todas las representaciones actuales del recurso de destino con la carga útil (payload) de la petición.
- Características: No es seguro pero es idempotente (si se repite, el recurso seguirá teniendo el mismo contenido final).
- Ejemplo: Reemplazar un archivo completo en un servidor o actualizar un perfil de usuario con todos sus campos.

DELETE (Eliminar - Delete):

- Función: Borra el recurso especificado.
- Características: No es seguro pero es idempotente (una vez eliminado el recurso, las peticiones siguientes no tendrán un efecto adicional, aunque el código de respuesta cambie).
- Ejemplo: Eliminar una foto o dar de baja una cuenta.

PATCH (Actualización parcial):

- Función: Aplica modificaciones parciales a un recurso.
- Características: No es seguro y no es idempotente (generalmente). Se usa para ser más eficiente que PUT al enviar solo los campos modificados.

La utilización del protocolo HTTPS simplemente añade una capa de seguridad (SSL/TLS) que cifra la comunicación completa (incluidos los métodos, cabeceras y cuerpo), protegiendo así los datos sensibles. Los métodos de petición subyacentes (GET, POST, etc.) siguen siendo los mismos.

Diferencia entre GET, POST, PUT y DELETE

https://www.youtube.com/watch?v=dXF3cc8mkHM

MDN Web Docs: Métodos de petición HTTP

https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Methods

AWS: HTTP y HTTPS: diferencia entre los protocolos de transferencia

https://aws.amazon.com/es/compare/the-difference-between-https-and-http/

4. Estudio sobre el concepto de URI (Identificador de Recursos Uniforme)/URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS.

URI es la categoría general para identificar recursos; **URL** es el mecanismo para localizar y acceder a ellos (usando protocolos como HTTP/HTTPS), y **URN** es el nombre persistente e independiente de la ubicación.

URI (Uniform Resource Identifier - Identificador Uniforme de Recursos)

El URI es el concepto fundamental y el término más amplio. Es una cadena de caracteres que sirve para identificar de forma única un recurso, ya sea físico (como un archivo) o abstracto (como una persona o un concepto).

- Utilidad: Proporcionar una identidad inequívoca a cualquier recurso en un contexto global.
- Estructura Genérica (Sintaxis RFC 3986):
  - esquema://autoridad]ruta[?consulta][#fragmento]
  - esquema (http, https, mailto, ftp, urn): Define el protocolo o la convención usada.
  - autoridad (www.ejemplo.com:8080): Identifica el servidor, a menudo el host y el puerto.
  - ruta (/documentos/index.html): Indica la jerarquía específica del recurso.
  - consulta (?id=123&sort=asc): Datos adicionales no jerárquicos (usados en búsquedas o formularios).
  - fragmento (#seccion5): Identificador de una parte específica del recurso principal.

URL (Uniform Resource Locator - Localizador Uniforme de Recursos)

El URL es el tipo de URI más conocido y utilizado. Su función no es solo identificar un recurso, sino también proporcionar la información necesaria para localizarlo y acceder a él mediante un mecanismo de red específico.

- Utilidad: Actuar como la dirección digital que permite a los navegadores y aplicaciones recuperar un recurso.
- Estructura: El URL es un URI que incluye necesariamente el componente de localización (protocolo y ubicación de red). Un URL siempre debe tener un esquema que implique un protocolo de acceso (como http, https, ftp, file, etc.).

URN (Uniform Resource Name - Nombre Uniforme de Recursos)

El URN es un tipo de URI que identifica un recurso por su nombre, de manera persistente e independiente de la ubicación. Es la respuesta a la pregunta "qué" es el recurso, no "dónde" está.

- Utilidad: Garantizar que el identificador siga siendo válido incluso si el recurso se mueve o deja de estar disponible en línea. Es crucial en biblioteconomía, estándares técnicos y archivos.
- Estructura: Siempre utiliza el esquema urn:, seguido de un identificador de espacio de nombres (NID) y una cadena específica (NSS).
  - urn: <NID> : <NSS>
  - Ejemplo: urn:ietf:rfc:3986 (Identifica el documento de estándar RFC 3986, sin decir dónde descargarlo).

RFC 3986 (Especificación URI): https://www.rfc-editor.org/rfc/rfc3986

MDN Web Docs: Guía de URLs: https://developer.mozilla.org/es/docs/Learn/Common_questions/What_is_a_URL

W3C: URIs, URLs, and URNs: Clarifications and Recommendations (en inglés): https://www.w3.org/TR/uri-clarification/

5. Modelo de desarrollo de aplicaciones multicapa – comunicación entre capas – componentes – funcionalidad de cada capa.

El **Modelo de Desarrollo de Aplicaciones Multicapa** (o Arquitectura de N Capas) es un patrón de diseño de software que organiza una aplicación en capas lógicas que se especializan en una función específica, lo que promueve la separación de responsabilidades, la escalabilidad y la mantenibilidad. La arquitectura más común es la de Tres Capas (3-Tier), aunque pueden existir más (N-Capas) dependiendo de la complejidad de la aplicación. Las tres capas fundamentales son:

- **Capa de Presentación**: Se encarga de la interacción con el usuario, presentando la información y capturando las entradas. Utiliza las tecnologías HTML, CSS, JavaScript y frameworks como React, Angular, Vue.js, etc. Es lo que se ejecuta en el navegador del usuario.

- **Capa de lógica de negocio o capa de aplicación**: Contiene las reglas de negocio que definen cómo se deben procesar los datos, qué operaciones son válidas y coordina la aplicación. Es el "cerebro" de la aplicación. Utiliza tecnologías como Java, Python, C#, Node.js, PHP, Ruby y frameworks como Spring Boot, Django, ASP.NET, Express.js. Se aloja en el servidor de aplicaciones.

- **Capa de acceso a datos**: Gestiona la persistencia de los datos, interactuando directamente con la base de datos (BD). Se encarga de almacenar y recuperar la información solicitada por la capa de negocio. Se usan sistema gestores de bases de datos como MySQL, PostgreSQL, Oracle, MongoDB. Se aloja en el servidor de bases de datos.

Comunicación entre Capas

La comunicación ideal sigue el principio de dependencia unidireccional y estratificación estricta (o cerrada):

- Flujo de Petición: La Capa de Presentación llama a la Capa de Lógica de Negocio, y esta a su vez llama a la Capa de Acceso a Datos.

- Flujo de Respuesta: La Capa de Acceso a Datos devuelve la información a la Capa de Lógica de Negocio, y esta la procesa y la envía de vuelta a la Capa de Presentación para que la muestre al usuario.

- Encapsulamiento: Una capa solo se comunica con la capa inmediatamente adyacente

(superior o inferior). Esto significa que la Capa de Presentación no puede acceder directamente a la Capa de Acceso a Datos; siempre debe pasar a través de la Capa de Lógica de Negocio.

Arquitectura de software - Arquitectura en capas https://www.youtube.com/watch?v=A8AJ_bb7Oec

6. Modelo de división funcional front-end / back-end para aplicaciones web.

El modelo separa la aplicación en dos grandes "puertas de acceso", cada una diseñada para satisfacer las necesidades y el nivel de acceso de diferentes audiencias:

- Front-end: La Puerta para el Usuario Normal: está optimizado y diseñado para el usuario final, independientemente de si está autenticado (ha iniciado sesión) o no.

Es todo lo relacionado con la **interfaz y la interacción directa** con el contenido. Esto incluye: * Navegación de contenido público (páginas de inicio, productos, blogs). * Procesos de compra o reserva. * Funcionalidad de perfil de usuario (ver/editar). * Búsqueda y filtrado de datos. Solo solicita y muestra datos. Nunca contiene lógica de programación. Cualquier cambio de estado crítico (ej. realizar una compra) se hace mediante llamadas API al Back-end.

- Back-end: El Centro de Control para Usuarios Especializados: no es solo el centro de lógica y datos, sino que también aloja las herramientas y la funcionalidad crítica necesaria para los roles administrativos y de gestión.

Orientado a Usuarios internos con roles privilegiados (Administrador, Censor, Editor, Moderador, Regulador, etc.).

Sirve par acciones que gestionan o modifican el sistema a nivel estructural o de datos. Esto incluye: * Creación, edición y eliminación de contenido (por publicadores). * Aprobación o rechazo de contenido (por censores/moderadores). * Gestión de permisos y roles de usuario (por administradores). * Análisis y reportes de datos a gran escala. * Lógica de negocio crítica (cálculo de nóminas, gestión de inventario).

La seguridad y la correcta separación se basan en cómo el Front-end se comunica con el Back-end:

Front-end (Usuario Normal): Solicita datos a través de la **API Pública**. Por ejemplo, pide los datos de un producto. El Back-end solo le devuelve los **datos necesarios**.

Usuarios Especializados: Utilizan un

(a menudo una aplicación separada o un dashboard) que se conecta a la API Privada/Administrativa del Back-end. Esta API comprueba rigurosamente el rol del usuario antes de permitir la ejecución de cualquier funcionalidad crítica (ej. la eliminación de una cuenta).

Esta división garantiza que las funcionalidades críticas para la gestión del sistema residan en el servidor (Back-end) y estén protegidas por su lógica de autorización, siendo inaccesibles y desconocidas para la mayoría de los usuarios.

¿Cuál es la diferencia entre el front end y back end en el desarrollo de aplicaciones?

https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/#:~:text=El%20desarrollo%20del%20front%20end%20se%20centra%20en%20crear%20interfaces,el%20desarrollo%20del%20front%20end.

7. Página web estática – página web dinámica – aplicación web – mashup .

La arquitectura web se divide por complejidad y rol de usuario: Estática (contenido fijo, solo Front-end), Dinámica (contenido variable por Back-end y base de datos), Aplicación Web (alta funcionalidad, interacción Front-end/Back-end intensa) y Mashup (integra datos de múltiples APIs externas).

Página Web Estática

Una página web estática es el tipo más simple, actuando como un documento digital fijo. El contenido se almacena en el servidor como archivos de texto simple (HTML, CSS) y se envía al navegador sin modificación alguna. No involucra lógica de negocio en el servidor ni bases de datos. Son ideales para sitios de una sola página, landings promocionales o documentación que no cambia.

Página Web Dinámica

Las páginas dinámicas utilizan un Back-end y una base de datos para generar contenido personalizado en el momento de la solicitud. Por ejemplo, al visitar un sitio de noticias, el servidor consulta la base de datos para obtener los titulares más recientes y los inserta en la plantilla HTML antes de enviarla al cliente. Esto permite la personalización, la interactividad básica (como buscar o filtrar) y la actualización frecuente de contenido.

Aplicación Web (WebApp)

Una aplicación web va más allá de mostrar información; está diseñada para que el usuario realice tareas y gestione datos. Se caracterizan por su alta interactividad y la sensación de ser un programa de software tradicional. La mayor parte de la lógica de presentación se ejecuta en el Front-end (a menudo como una SPA - Single Page Application), comunicándose con el Back-end a través de APIs para el procesamiento de datos, autenticación y lógica de negocio compleja.

Mashup

Un mashup es una aplicación web que adquiere su valor al combinar servicios y datos de múltiples fuentes externas. Extrae información (a través de APIs) de servicios de terceros (ej. Google Maps, Twitter, etc) y los integra en una única interfaz coherente. El objetivo es crear un servicio nuevo y funcionalmente rico a partir de componentes existentes, como una inmobiliaria que muestre propiedades junto a mapas y valoraciones de transporte.

Mashup

https://es.wikipedia.org/wiki/Mashup_(aplicaci%C3%B3n_web_h%C3%ADbrida)

8. Componentes de una aplicación web.

Los componentes esenciales de una aplicación web moderna se agrupan en cuatro capas: Front-end (lo que el usuario ve: HTML, CSS, JavaScript), Back-end (el motor de lógica: Servidor de Aplicaciones, Framework), Datos (Base de Datos) y la API (el canal de comunicación entre el Front-end y el Back-end).

La Capa del Cliente (Front-end)

Es el lado que se ejecuta en el navegador del usuario y es responsable de la Experiencia del Usuario (UX/UI) y la Presentación.

- **HTML** (HyperText Markup Language): Proporciona la estructura y el contenido semántico de la página (el esqueleto).
- **CSS** (Cascading Style Sheets): Define el estilo visual de la estructura (diseño, colores, tipografía, la "piel").
- **JavaScript**: Agrega la interactividad, maneja la lógica de presentación del lado del cliente y realiza peticiones asíncronas al servidor (el "cerebro" del cliente).
- **Frameworks y Librerías**: Herramientas para construir interfaces complejas de manera eficiente (React, Angular, Vue.js).

La Capa del Servidor (Back-end)

Es el lado que se ejecuta en el servidor y es responsable de la Lógica de Negocio, la seguridad y la coordinación de datos.

- Servidor Web: Software que gestiona las peticiones HTTP/HTTPS (ej. Nginx, Apache).

- Servidor de Aplicaciones: El entorno donde se ejecuta el código que procesa las peticiones, implementa la lógica de negocio y se comunica con la base de datos (utilizando lenguajes como Python, Java, Node.js, con frameworks como Django, Spring, Express).

- **Lógica de Negocio**: El conjunto de reglas, cálculos y algoritmos que definen el comportamiento único de la aplicación (ej. validar un pago, calcular impuestos).

La Interfaz de Conexión (API)

Es el puente que permite la comunicación entre el Front-end y el Back-end.

API (Application Programming Interface): Un conjunto de reglas bien definidas (un "contrato") que permite a los dos sistemas interactuar. Generalmente, se implementan como APIs RESTful que intercambian datos en formato JSON.

La Capa de Datos (Base de Datos)

El sistema de almacenamiento donde residen los datos de la aplicación (usuarios, productos, transacciones). Pueden ser relacionales (MySQL, PostgreSQL) o no relacionales (MongoDB, Cassandra).

9. Programas ejecutados en el lado del cliente y programas ejecutados en el lado del servidor - lenguajes de programación utilizados en cada caso.

Programas Ejecutados en el Lado del Cliente (Front-end)

Estos programas se descargan y ejecutan directamente en el navegador web del usuario. Su función principal es gestionar la interfaz, la interactividad y la experiencia del usuario (UX/UI).

- JavaScript (JS): Es el único lenguaje de programación que el navegador ejecuta de forma nativa para proporcionar interactividad. Permite:

Manipular la estructura y el estilo de la página (el DOM).
Manejar eventos (clics, entradas de teclado).
Realizar peticiones al servidor (AJAX/Fetch) sin recargar la página.

- **HTML** (HyperText Markup Language): Aunque es un lenguaje de marcado, define la estructura y el contenido.

- **CSS** (Cascading Style Sheets): Aunque es un lenguaje de hojas de estilo, define la presentación y el diseño.

- Lenguajes/Frameworks Transcompilados: Tecnologías que se escriben en un lenguaje y se convierten a JavaScript para ser ejecutadas en el navegador (ej. TypeScript, Dart/Flutter, Svelte, JSX).

Programas Ejecutados en el Lado del Servidor (Back-end)

Estos programas se ejecutan en un servidor remoto y son responsables de la lógica de negocio, la autenticación, la seguridad y la gestión de datos. Reciben peticiones del Front-end, las procesan, interactúan con la base de datos y devuelven una respuesta (normalmente JSON). Utiliza una amplia variedad de lenguajes. La elección depende de la escalabilidad, el rendimiento y las preferencias del equipo.

- **Python**: Popular por su legibilidad, rapidez de desarrollo y frameworks robustos (Django, Flask). Ideal para machine learning y APIs complejas.
- **Java**: Conocido por su rendimiento, escalabilidad y uso en sistemas empresariales a gran escala (Spring Framework).
- **JavaScript** (Node.js): Permite ejecutar JavaScript fuera del navegador. Es muy popular por su arquitectura asíncrona y su eficiencia (Express.js). Permite que una persona o equipo trabaje con el mismo lenguaje en ambos lados.
- **PHP**: Uno de los lenguajes originales del desarrollo web. Aún es muy utilizado por plataformas como WordPress y frameworks modernos como Laravel.
- **Ruby**: Elegante y popular por su framework Ruby on Rails, que prioriza la convención sobre la configuración.
- **C# (.NET)**: Desarrollado por Microsoft, ampliamente utilizado en entornos Windows y empresariales (ASP.NET).

- **Go (Golang)**: Destaca por su alta concurrencia y velocidad, ideal para microservicios y sistemas de alto rendimiento.

Visión General Cliente-Servidor

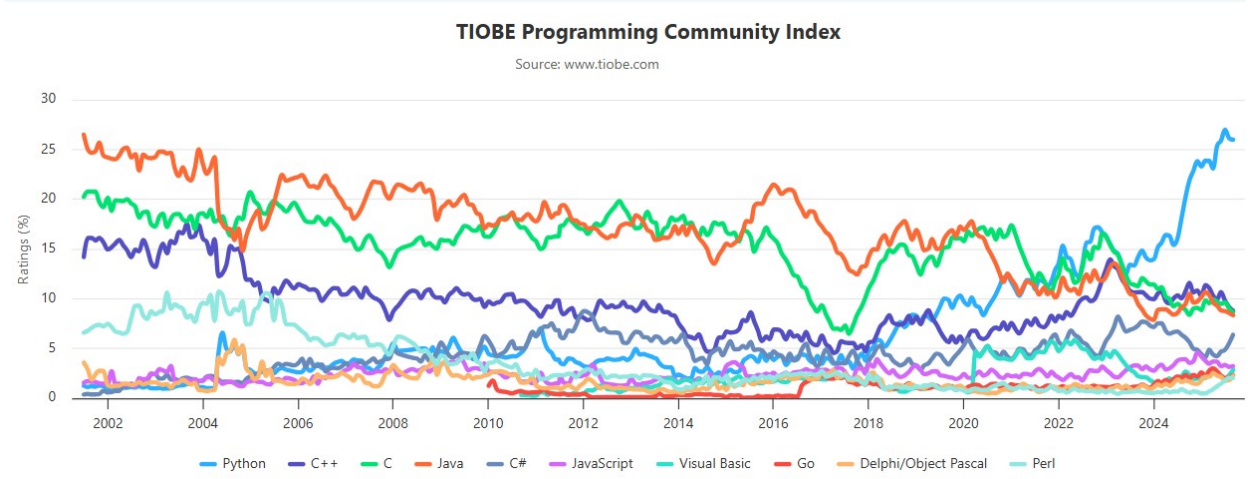
https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/First_steps/Client-Server_overview

Conoce los lenguajes de backend más demandados

https://teclab.edu.ar/tecnologia-y-desarrollo/lenguajes-de-backend-mas-demandados/

10. Lenguajes de programación utilizados en el lado servidor de una aplicación web (características y grado de implantación actual).

El lado servidor (Back-end) utiliza lenguajes que se centran en la **lógica de negocio y el rendimiento**. Los más usados actualmente son **JavaScript (Node.js)** (rápido, moderno), **Python** (fácil, *frameworks* robustos), **Java** (escalable, empresarial) y **PHP** (amplia base instalada, especialmente CMS).



JavaScript (Node.js)

Permite utilizar el mismo lenguaje tanto en el Front-end como en el Back-end (Full-Stack JavaScript). Es rápido, altamente escalable y no bloqueante (maneja múltiples peticiones simultáneamente sin esperar). Usa el runtime Node.js y el framework Express.js.

- Implantación Actual: Muy alta. Es uno de los lenguajes de crecimiento más rápido y dominante en la web moderna, especialmente en el desarrollo de APIs RESTful y microservicios.

Python

Lenguaje de sintaxis clara y legible. Famoso por su velocidad de desarrollo y sus potentes frameworks de desarrollo web (Django, Flask). Su fortaleza en la ciencia de datos (ML/AI) lo integra frecuentemente en aplicaciones web que requieren análisis complejos.

- Implantación Actual: Extremadamente alta. Es omnipresente, no solo en la web, sino también en el análisis de datos. Muy demandado por startups y empresas que buscan rapidez

y robustez.

Java

Lenguaje orientado a objetos, conocido por su rendimiento, escalabilidad y estabilidad. Es la columna vertebral de muchas aplicaciones empresariales grandes y sistemas heredados. Se apoya en el poderoso framework Spring (especialmente Spring Boot).

- Implantación Actual: Alta y estable. Es el estándar en grandes corporaciones, banca, seguros y sistemas que requieren una alta disponibilidad y tolerancia a fallos.

PHP

Lenguaje diseñado específicamente para el desarrollo web. Es conocido por su facilidad de aprendizaje y su enorme ecosistema. Hoy en día, frameworks como Laravel han modernizado y mejorado significativamente su rendimiento y seguridad.

- Implantación Actual: Masiva. A pesar de la competencia, sigue siendo el lenguaje más utilizado por el número de sitios web que lo usan, ya que es la base de sistemas de gestión de contenido (CMS) como WordPress y Drupal.

TIOBE Index https://www.tiobe.com/tiobe-index/

11. Características y posibilidades de desarrollo de una plataforma XAMPP.

XAMPP es un entorno de desarrollo web local y gratuito que permite ejecutar aplicaciones dinámicas sin necesidad de un servidor externo. Sus principales características son su **facilidad de instalación** (*cross-platform*) y su pila tecnológica basada en **Apache, MariaDB/MySQL, PHP y Perl**. Permite desarrollar sitios dinámicos, CMS y APIs de Back-end.

Es un paquete de *software* libre (de código abierto) que instala y configura de manera sencilla un entorno de servidor web local completo en tu propio ordenador. El nombre es un acrónimo de sus componentes clave, con la 'X' representando que es multiplataforma.

Características Principales de XAMPP

- **Multiplataforma** (X): Se instala y funciona de manera idéntica en los sistemas operativos más comunes: Windows, macOS y Linux. Esto garantiza que el código desarrollado en un entorno se comporte de manera consistente.
- **Servidor Apache** (A): Es el servidor web más utilizado del mundo. XAMPP lo incluye para que pueda interpretar y servir páginas web locales desde una carpeta específica (generalmente htdocs).
- **Base de Datos** (M): Utiliza MariaDB (o MySQL, su predecesor original). Permite crear bases de datos relacionales necesarias para cualquier aplicación web dinámica (ej. almacenar usuarios, productos, publicaciones de blog).
- Lenguajes de Scripting (P's):

PHP: El principal lenguaje para el desarrollo del lado del servidor. Permite procesar formularios, interactuar con la base de datos y generar contenido HTML dinámico.

Perl: Un lenguaje de scripting versátil, aunque menos dominante en web que PHP, incluido para tareas de automatización y scripting de sistema.

- Entorno de Gestión (**phpMyAdmin**): Incluye una interfaz gráfica web que facilita enormemente la gestión de las bases de datos MariaDB/MySQL sin necesidad de usar comandos de consola.

Posibilidades de Desarrollo con XAMPP

XAMPP está optimizado para cualquier tipo de desarrollo que utilice la pila LAMP/WAMP (Linux/Windows, Apache, MySQL, PHP). Sus principales posibilidades son:

- **Desarrollo de Sitios Web Dinámicos:** Es la plataforma ideal para construir cualquier sitio que requiera una base de datos, desde un formulario de contacto que guarda datos hasta un sitio web corporativo completo.
- **Implementación de CMS Populares**: Permite instalar y desarrollar localmente sobre los CMS más usados del mundo, como WordPress, Joomla! y Drupal, antes de desplegarlos en un servidor real.
- **Pruebas de Back-end y APIs**: Puedes crear y probar localmente la lógica del Back-end, incluyendo APIs sencillas basadas en PHP para que el Front-end (desarrollado en el navegador) pueda consumir datos de prueba de forma segura.
- **Aprendizaje y Prototipado**: Es la herramienta de inicio preferida para estudiantes y desarrolladores que quieren aprender desarrollo Back-end (PHP) y bases de datos, ya que su configuración es trivial.
- **Integración de Aplicaciones Existentes:** Permite ejecutar localmente aplicaciones que ya están en producción para realizar mantenimiento, pruebas de nuevas funcionalidades (testing) o depuración de errores (debugging).

XAMPP Documentation https://www.apachefriends.org/docs/

12. En que casos es necesaria la instalación de la máquina virtual Java (JVM) y el software JDK en el entorno de desarrollo y en el entorno de explotación.

La instalación de la Máquina Virtual Java (JVM) es necesaria en cualquier máquina que deba ejecutar una aplicación Java. El Java Development Kit (JDK) solo es necesario en las máquinas utilizadas para compilar, depurar o desarrollar software Java.

Entorno de Desarrollo

El JDK (Java Development Kit) sí **es imprescindible**. Contiene herramientas esenciales como el compilador (javac) para transformar el código fuente (.java) en bytecode (.class) y herramientas de depuración y creación de packages (como jar).

La JVM (Java Virtual Machine) también **es imprescindible**. Es necesaria para ejecutar el bytecode compilado en cualquier fase de prueba. El JDK ya incluye una JVM (a través del JRE - Java Runtime Environment).

Entorno de Explotación

JDK (Java Development Kit) **no es necesario**, el código ya está compilado y las herramientas de desarrollo no son necesarias. Instalar el JDK en producción puede ocupar espacio innecesario y, en algunos casos, ser un riesgo de seguridad.

JVM (Java Virtual Machine) sí **es imprescindible.** Es el componente esencial para cargar y ejecutar el bytecode (.class) de la aplicación Java en el sistema operativo.

13. IDE más utilizados (características y grado de implantación actual).

Visual Studio

Visual Studio es un IDE multiplataforma desarrollado por Microsoft. Su lanzamiento se remonta a 1997, logrando una popularidad sólida a día de hoy.

Soporte para una amplia gama de lenguajes de programación.

Mejor integración con otros productos de Microsoft.

Creación de aplicaciones nativas e híbridas.

IntelliCode con Inteligencia Artificial incorporada.

Eclipse

Eclipse es un IDE multiplataforma y multilenguaje utilizado popularmente para lenguaje Java.

Ampliación de funciones mediante plugins.

Plataforma RCP.

Gestión basada en proyectos.

Depuración de código.

NetBeans

NetBeans es un IDE orientado principalmente a programar en Java. Sus mejoras continuas y la flexibilidad de sus funciones lo convierten en una de las mejores opciones en el desarrollo de aplicaciones.

Su gestión de ventanas

Desarrollo de aplicaciones integral

Asistencia inteligente

Amplia gama de herramientas

Los 20 mejores editores de codigo y HTML e IDE https://www.arsys.es/blog/mejores-editores-html-ide

Porcentajes de Implantación de IDEs (Datos de la Encuesta Stack Overflow más Reciente)

El mercado está claramente dominado por las herramientas de Microsoft y la suite JetBrains, con **Visual Studio Code** como líder indiscutible.

IDE / Editor	Tipo	Porcentaje de Uso (Aproximado)
Visual Studio Code (VS Code)	Editor de código / IDE ligero	**~75 - 80%**
Visual Studio	IDE completo (Microsoft)	**~30 - 35%**
IntelliJ IDEA	IDE (JetBrains)	**~30%**
Eclipse	IDE (Open Source)	**~15 - 20%**
NetBeans	IDE (Open Source)	**~5 - 10%**

Nota Importante: *Los encuestados pueden seleccionar* ***múltiples IDEs*** *que utilizan, por lo que los porcentajes no suman 100%.*

https://survey.stackoverflow.co/2024/technology#3-integrated-development-environment

14. Servidores HTTP /HTTPS más utilizados (características y grado de implantación actual).

Según datos recientes (W3Techs, octubre de 2025), el mercado se distribuye de la siguiente manera, con **Nginx y Apache** como líderes.

Servidor Web	Cuota de Mercado (Aproximada)	Enfoque Principal
Nginx	**~33.4%**	Alto rendimiento, conexiones concurrentes, *proxy* inverso.
Apache HTTP Server	**~25.5%**	Versatilidad, contenido dinámico, madurez, comunidad.
Cloudflare Server	**~24.4%**	CDN, seguridad (WAF), optimización de velocidad.
4. **LiteSpeed**	**~14.8%**	Velocidad pura, eficiencia con PHP (WordPress, etc.).
5. **Microsoft-IIS**	**~3.7%**	Entornos Windows y aplicaciones .NET.

Nginx (Engine-X)

Arquitectura Event-Driven (Basada en eventos): A diferencia de la arquitectura basada en procesos de Apache, Nginx utiliza un único proceso con múltiples subprocesos para

gestionar miles de conexiones concurrentes de manera eficiente.

Bajo Consumo de Recursos: Es muy ligero en memoria, lo que lo hace ideal para servidores con alto tráfico.

Proxy Inverso y Balanceador de Carga: Su uso más común en sitios grandes es actuar como front-end rápido (proxy inverso) que distribuye las peticiones a servidores de aplicaciones (como Apache Tomcat, Node.js, etc.) en el back-end.

Seguridad (HTTPS): Soporta SSL/TLS de forma eficiente. Su diseño como proxy lo hace excelente para la terminación SSL.

Apache HTTP Server

Modularidad y Madurez: Es uno de los servidores más antiguos y estables, con una comunidad masiva y una vasta colección de módulos (mod_rewrite, mod_security, etc.) que extienden su funcionalidad.

Arquitectura Basada en Procesos: Utiliza Multi-Processing Modules (MPMs) para gestionar las conexiones (cada conexión puede requerir más recursos).

Versatilidad: Es ideal para entornos que ejecutan múltiples lenguajes dinámicos (PHP, Python) a través de módulos dedicados.

Seguridad (HTTPS): Soporta SSL/TLS y su gran cantidad de módulos de seguridad lo hacen altamente configurable.

Sigue siendo una opción muy popular para servidores compartidos y proyectos con necesidades dinámicas que no requieren el altísimo rendimiento de Nginx para conexiones concurrentes.

https://w3techs.com/

15. Apache HTTP vs Apache Tomcat

La principal diferencia radica en su propósito y funcionalidad principal:

- Apache **HTTP** Server (o httpd) es un Servidor Web tradicional.
- Apache **Tomcat** es un Contenedor de Servlets/Servidor de Aplicaciones Java.

Característica	Apache HTTP Server	Apache Tomcat
Función Principal	Servidor web que sirve **contenido estático** (archivos HTML, imágenes, CSS, JavaScript).	Servidor de aplicaciones/Contenedor de Servlets que ejecuta **código Java dinámico** (Servlets y JSP).
Contenido que Sirve	Estático y dinámico (usando módulos como PHP, Python, Perl, etc.).	Dinámico basado en Java. Puede servir estático, pero no es su fuerte.
Lenguaje Base	Escrito principalmente en **C**.	Escrito en **Java**.
Requisitos	No requiere la máquina virtual de Java (JVM).	**Requiere** una instalación de Java Development Kit (JDK) para funcionar.
Velocidad	Generalmente **más rápido** y eficiente para	Más lento que `httpd` para

Característica	Apache HTTP Server	Apache Tomcat
(Estático)	servir archivos estáticos.	servir archivos estáticos.
Escenario de Uso	Alojar sitios web estáticos, servir contenido multimedia, actuar como *proxy* inverso o balanceador de carga.	Alojar aplicaciones web empresariales construidas con tecnologías Java (Spring, JSPs, Servlets).
Trabajo Conjunto	Comúnmente se usan **juntos** (Apache HTTP Server recibe todas las peticiones, sirve el contenido estático directamente y reenvía las peticiones de contenido dinámico a Tomcat).	

Tomcat vs Apache: What's the difference?
https://www.youtube.com/watch?v=XABDkzxA6hM

16. Navegadores HTTP /HTTPS más utilizados (características y grado de implantación actual).

El grado de implantación es abrumadoramente favorable a Google Chrome, seguido por Safari, impulsado por el ecosistema Apple.

Posición	Navegador	Cuota de Mercado (Aprox.)	Motor de Renderizado
1.	**Google Chrome**	**65% - 70%**	Chromium (Blink)
2.	**Apple Safari**	**17% - 19%**	WebKit
3.	**Microsoft Edge**	**4% - 5%**	Chromium (Blink)
4.	**Mozilla Firefox**	**2% - 3%**	Gecko
5.	**Samsung Internet**	**2% - 3%**	Chromium (Blink)
6.	**Opera**	**1% - 2%**	Chromium (Blink)

https://gs.statcounter.com/browser-market-share

Google Chrome (El Líder)

- **Motor: Chromium** (Blink). Al ser el motor de la mayoría de los navegadores modernos, garantiza una alta compatibilidad con todos los estándares web.
- **Velocidad y Sincronización:** Es conocido por su **velocidad** de carga y la **sincronización** fluida de datos (historial, contraseñas, pestañas) a través de la cuenta de Google en cualquier plataforma (Windows, Mac, Linux, iOS, Android).
- **Extensiones:** Posee la **mayor biblioteca de extensiones** del mercado.
- **Desventaja:** Históricamente criticado por su **alto consumo de memoria RAM**.

2. Apple Safari (El Ecosistema)

- **Motor: WebKit**.
- **Integración Apple:** Es la opción por defecto en todos los dispositivos Apple (iPhone, iPad, Mac), lo que impulsa su alta cuota de mercado, especialmente en móviles.
- **Privacidad y Rendimiento:** Se destaca por su **eficiencia energética** (ideal para portátiles y móviles) y funciones de **privacidad** robustas, como el Bloqueo Inteligente de Rastreadores (ITP).

3. Microsoft Edge (El Renacido)

- **Motor: Chromium** (Blink). Tras abandonar su motor EdgeHTML, adoptó Chromium en 2020.
- **Integración Windows:** Ofrece una **excelente integración con Windows** (ej. colecciones, modo de lectura, búsqueda integrada).
- **Funciones AI:** Microsoft ha incorporado funciones de **IA (Copilot/ChatGPT)** para resumir páginas web y búsquedas dentro del navegador.
- **Rendimiento:** Es generalmente más ligero en recursos que Chrome.

4. Mozilla Firefox (El Campeón de la Privacidad)

- **Motor: Gecko**. Es el único gran navegador que utiliza un motor independiente (no basado en Chromium), promoviendo la diversidad en el desarrollo web.
- **Código Abierto y Privacidad:** Es la opción predilecta para usuarios que valoran el **código abierto** y la **privacidad**. Ofrece funciones avanzadas de **protección contra rastreo** y DNS sobre HTTPS (DoH) por defecto.
- **Personalización:** Altamente personalizable con temas y una sólida biblioteca de extensiones.

17. Generadores de documentación HTML (PHPDoc): PHPDocumentor, ApiGen, …

18. Repositorios de software – sistemas de control de versiones: GIT , CVS, Subversion, …

19. Propuesta de configuración del entorno de desarrollo para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USED y xxx-WXED.

20. Propuesta de configuración del entorno de explotación para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USEE.

21. Realizar un estudio sobre los siguientes conceptos y su relación con el desarrollo de aplicaciones web: CMS – Sistema de gestión de contenidos ERP – Sistema de planificación de los recursos empresariales

22. Elegir y realizar un estudio y una presentación para la exposición del trabajo sobre una de las siguientes arquitecturas de desarrollo de Aplicaciones Web:

- MEAN (con MongoDB y con MySQL)
- Java EE vs Spring
- Microsoft .NET
- Angular 7
- Symfony
- Laravel
- CakePHP
- CodeIgniter

GLOSARIO DE TÉRMINOS RELACIONADOS CON DWES

Estudios propuestos para el módulo Desarrollo Web en Entorno Servidor

Contenidos y la diferencia entre los módulos que tienes en este curso.

Protocolos TCP/IP. Socket.

Protocolos TCP/IP

TCP/IP es el conjunto de protocolos de comunicación que permite que las computadoras se conecten e intercambien datos. Es la base de Internet. El nombre es la abreviatura de sus dos protocolos más importantes: el Protocolo de Control de Transmisión (TCP) y el Protocolo de Internet (IP). Se suele representar como una arquitectura de capas (similar al modelo OSI, pero simplificado).

- **Protocolo IP**: Se encarga del direccionamiento y enrutamiento de los paquetes de datos (llamados datagramas) de origen a destino a través de diferentes redes.

- **Protocolo TCP**: Se encarga de la entrega confiable de datos entre dos aplicaciones. En protocolos que requieren alta fiabilidad, como HTTP (navegación web), FTP (transferencia de archivos) y SMTP (correo electrónico).

Socket (Toma de Comunicación)

Un Socket es el punto final lógico de una conexión de comunicación a través de una red. Es una abstracción de programación que permite a las aplicaciones enviar y recibir datos

utilizando los protocolos de red (como TCP o UDP/IP). Se define típicamente por una tupla de tres elementos.

Socket = (Protocolo,Dirección IP,Número de Puerto)

- Protocolo: Suele ser TCP (para sockets orientados a conexión) o UDP (para sockets sin conexión).

- **Dirección IP**: La dirección del host donde reside la aplicación (por ejemplo, el servidor web).

- **Número de Puerto:** Un número entero de 16 bits (del 0 al 65535) que identifica la aplicación o servicio específico dentro de ese host. Por ejemplo, el puerto 80 es para HTTP, el 443 para HTTPS y el 22 para SSH.

Protocolo HTTP / HTTPS

HTTP es un protocolo de la capa de aplicación diseñado para la comunicación cliente-servidor entre un navegador web (cliente) y un servidor web. Permite la transferencia de datos, principalmente documentos HTML (páginas web), imágenes, videos y otros recursos.

Opera sobre TCP. Un cliente (navegador) envía una **solicitud HTTP** a un servidor, y el servidor responde con una **respuesta HTTP** (que incluye el recurso solicitado y un código de estado, como 200 OK). Utiliza el puerto 80 por defecto.

HTTPS cifra la comunicación entre el navegador y el servidor, asegurando tres pilares de la seguridad. Utiliza el puerto 443 por defecto.

- **Cifrado** (Encryption): La información intercambiada se codifica para que sea ilegible si es interceptada por terceros.

- **Integridad** (Integrity): Garantiza que los datos no sean modificados o corrompidos durante la transferencia.

- **Autenticación** (Authentication): Verifica la identidad del servidor al que el cliente se está conectando. Esto se logra mediante Certificados SSL/TLS emitidos por una Autoridad de Certificación (CA). Si el certificado es válido, el navegador confía en que está hablando con el servidor legítimo.

HTML

Hace referencia al lenguaje de marcado utilizado en la creación de páginas web. Define una estructura básica y un código para la presentación de contenido de una página web, que incluye texto, imágenes, videos, juegos, entre otros elementos.

XML

Significa "Lenguaje de Marcado Extensible" . XML, similar a HTML, es un lenguaje de programación para estructurar y presentar datos en un formato de texto formateado. Fue desarrollado originalmente para permitir el intercambio de datos independientemente de los sistemas operativos y lenguajes de programación.

JSON

Acrónimo de JavaScript Object Notation, es un formato de texto sencillo para el intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Se refiere al hecho de insertar diferentes tipos de multimedia dentro de una página web.

Lenguajes de programación embebidos en HTML

Se refiere al hecho de insertar diferentes tipos de código dentro de una página web. La forma de hacerlo es mediante una etiquetas según el lenguaje, en nuestro caso las más utilizadas serán <style> para CSS, <Script> para javascript, <?php ?> para php.

Arquitecturas de desarrollo web

Se refiere a la estructura y organización de un sitio web, así como a las tecnologías y métodos utilizados en su creación. Este concepto abarca tanto el diseño visual como la funcionalidad, el rendimiento y la escalabilidad del sitio.

Framework de desarrollo Web

Un framework es un conjunto de herramientas, código y estructuras predefinidas que facilitan el desarrollo de software. Funciona como una base que ayuda a los programadores a construir aplicaciones sin empezar desde cero, ahorrando tiempo y esfuerzo.

En términos sencillos, es como un kit de construcción: en lugar de crear cada pieza desde el principio, el framework ya proporciona bloques listos para ensamblar. Esto permite enfocarse en las funciones específicas de la aplicación sin preocuparse por la parte técnica básica.

ERP

Es un sistema de software que ayuda a las organizaciones a optimizar sus procesos de negocio centrales —incluyendo finanzas, RR. HH., fabricación, cadena de suministro, ventas y procurement— con una visión unificada de la actividad y una única fuente de verdad.

CMS

Es un programa informático que permite la creación y administración de contenidos digitales, principalmente en páginas web, por parte de los administradores, editores, participantes y demás usuarios. Cuenta con una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio web. El sistema permite manejar de manera independiente el contenido y el diseño.

PHP

Es un lenguaje de programación interpretado del lado del servidor y de uso general que se adapta especialmente al desarrollo web. En un servidor web, el resultado del código PHP interpretado y ejecutado formaría la totalidad o parte de una respuesta HTTP al cliente.

IDE

Entorno de desarrollo integrado, es una aplicación informática que proporciona servicios integrales, las partes principales son el editor de código fuente, herramientas de construcción automática y depuración, herramientas para control de base de datos, control de versiones, etc.

Navegador

Es un software, aplicación o programa que permite el acceso a la Web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser vistos.

Repositorio

Es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos

científicos, conjuntos de datos o software.

Entorno de Desarrollo

Es un conjunto de procedimientos y herramientas que se utilizan para desarrollar un código fuente o programa. Aquí es donde el desarrollador prueba el código y comprueba si la aplicación se ejecuta correctamente con ese código. Una vez que la implementación ha sido probada y el desarrollador considera que el código trabaja de forma correcta, la aplicación se mueve entonces al entorno de explotación.

Entorno de Explotación o Producción

Es el entorno real en el que se ejecuta el software y se utiliza por los usuarios finales y por los clientes del software. Este entorno cuenta con una alta estabilidad y un rendimiento óptimo para garantizar una experiencia fluida a los usuarios finales.

Gestión de la configuración. Control de cambios. Mantenimiento de la aplicación.

Gestión de la Configuración

El proceso de identificar, documentar y mantener el estado funcional y físico de los componentes de la aplicación y la infraestructura (servidores, bases de datos, código fuente, archivos de configuración) a lo largo del tiempo.

El propósito es asegurar que se sabe QUÉ se tiene, CÓMO está configurado y que se puede reproducir la aplicación en cualquier momento y entorno.

Control de Cambios

El procedimiento sistemático y formal para solicitar, evaluar, priorizar, aprobar o rechazar, planificar e implementar cualquier modificación en un elemento de la configuración de la aplicación (código o infraestructura).

El propósito es garantizar que solo se realizan cambios necesarios y autorizados, minimizando el riesgo de fallos e interrupciones en el servicio de la aplicación.

Mantenimiento de la Aplicación

El conjunto de actividades continuas realizadas después del despliegue inicial para asegurar que la aplicación siga funcionando correctamente, satisfaciendo las necesidades del usuario y adaptándose al entorno cambiante.

Incluye corrección de errores (Correctivo), adaptación a nuevos entornos (Adaptativo), mejora de la funcionalidad y rendimiento (Perfectivo) y prevención de fallos (Preventivo).

Web Services

Es un servicio que hace posible la comunicación de máquina a máquina y el intercambio de datos entre aplicaciones a través de una red de Internet.

Gracias a los web service es posible realizar una gran cantidad de interacciones cotidianas entre aplicaciones, incluso sin que te des cuenta de ello. Por ejemplo, es necesario el uso de un web service al conectar la información de tu cuenta de Facebook con un juego que recién descargaste, al igual que para utilizar la información de inicio de sesión de Google, e incluso para abrir una nueva cuenta en otra aplicación sin rellenar el formulario.

AJAX

Significa JavaScript asíncrono y XML (Asynchronous JavaScript and XML). Es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona, procesando cualquier solicitud al servidor en segundo plano.

Desarrollo de aplicaciones multicapa. Estrategias de diseño de aplicaciones Web.

Consiste en la segmentación lógica de una aplicación en unidades funcionales separadas, denominadas capas. Cada capa tiene una misión específica y bien definida, y la comunicación entre ellas suele estar restringida, generalmente permitiendo solo interacciones con las capas adyacentes para mantener la independencia funcional de las capas entre sí.

El modelo más común para aplicaciones web es el de tres capas:

- **Capa de Presentación**: Se encarga de la interacción con el usuario, presentando la información y capturando las entradas. Utiliza las tecnologías HTML, CSS, JavaScript y frameworks como React, Angular, Vue.js, etc. Es lo que se ejecuta en el navegador del usuario.

- **Capa de lógica de negocio o capa de aplicación**: Contiene las reglas de negocio que definen cómo se deben procesar los datos, qué operaciones son válidas y coordina la aplicación. Es el "cerebro" de la aplicación. Utiliza tecnologías como Java, Python, C#, Node.js, PHP, Ruby y frameworks como Spring Boot, Django, ASP.NET, Express.js. Se aloja en el servidor de aplicaciones.

- **Capa de acceso a datos**: Gestiona la persistencia de los datos, interactuando directamente con la base de datos (BD). Se encarga de almacenar y recuperar la información solicitada por la capa de negocio. Se usan sistema gestores de bases de datos como MySQL, PostgreSQL, Oracle, MongoDB. Se aloja en el servidor de bases de datos.

El diseño de aplicaciones web, especialmente aquellas con arquitectura multicapa, se guía por principios y patrones que buscan optimizar la experiencia de usuario y la calidad del software. Más allá del modelo básico de capas, se emplean patrones para organizar el código dentro de ellas, algunos ejemplos son:

- **MVC** (Modelo-Vista-Controlador): Un patrón muy popular, especialmente en la Capa de Presentación y de Negocio. El **modelo** representa los datos y la lógica que los manipula. La **vista** muestra la interfaz de usuario que presenta los datos. El **controlador** actúa como intermediario, recibiendo peticiones, llamando al Modelo y seleccionando la Vista a mostrar.

- **Diseño Web Adaptable** (Responsive Web Design): Esencial para garantizar que la interfaz de usuario (Capa de Presentación) funcione correctamente en cualquier dispositivo (móvil, tableta, escritorio). Se suele usar un enfoque Mobile First (diseñar primero para la pantalla más pequeña).

Arquitectura multicapa https://es.wikipedia.org/wiki/Arquitectura_multicapa

Arquitectura de tres niveles https://www.ibm.com/mx-es/think/topics/three-tier-architecture

Aplicaciones basadas en microservicios

Es un estilo de arquitectura para desarrollar aplicaciones. Gracias a los microservicios, una aplicación grande puede separarse en partes independientes más pequeñas, cada una con su propio dominio de responsabilidad. Para servir una única solicitud de usuario, una

aplicación basada en microservicios puede llamar a muchos microservicios internos con los que preparar su respuesta.

SaaS: Sofware as a Service

Es un modelo de distribución de software donde el soporte lógico y los respectivos datos que maneja se alojan en los servidores de un proveedor, cuyo acceso es a través de internet. El proveedor no solo proporciona el hardware, sino también el software correspondiente. El cliente, por su parte, puede utilizar las diferentes funciones del software sin más preámbulos.

El correo electrónico, las redes sociales y las soluciones de almacenamiento de archivos en la nube (como Dropbox o Box) son ejemplos de aplicaciones SaaS que la gente usa todos los días en su vida personal.

Control de acceso a la aplicación web o los Web Services

El control de acceso es un proceso de seguridad que permite gestionar quién tiene autorización para acceder a datos y recursos en una organización. Este proceso implica dos componentes principales: autenticación, que verifica la identidad del usuario, y autorización, que determina el nivel de acceso permitido.

Validación de entrada de datos a una aplicación Web

Es el proceso de verificar y filtrar la información que ingresa a un sistema informático, ya sea a través de formularios web, aplicaciones o cualquier otro tipo de entrada de datos. Su objetivo principal es prevenir posibles ataques cibernéticos, como inyecciones de código malicioso o vulnerabilidades de seguridad, que podrían comprometer la integridad de los sistemas y la privacidad de los usuarios. La validación de entrada es esencial para mitigar estos riesgos y proteger la infraestructura tecnológica de posibles amenazas.

Posicionamiento de una aplicación Web

Se refiere al conjunto de estrategias y técnicas utilizadas para mejorar la visibilidad y el ranking de un sitio web en los resultados de búsqueda de los motores de búsqueda, como Google

Historia, situación actual y evolución del diseño de aplicaciones Web

Historia del Diseño Web: Evolución, Innovaciones y Tendencias Clave - TechDomotic

Filosofías de desarrollo del software

Las filosofías de desarrollo del software abarcan una variedad de enfoques y metodologías que buscan optimizar la creación y entrega de software. Algunas de las más destacadas incluyen:

- Desarrollo guiado por pruebas (TDD): Involucra escribir pruebas primero y refactorizar el código para mejorar la calidad y la velocidad del desarrollo.
- Desarrollo de diseño dirigido (D3): Un proceso ágil que trabaja en conjunto con SCRUM y XP para crear requerimientos innovadores y soluciones óptimas.
- Software libre y de código abierto: Se refiere a software que está licenciado de tal manera que los usuarios pueden modificarlo y compartirlo libremente.
- Software educativo: Diseñado para enseñar y aprender, permite el desarrollo de habilidades programáticas y la enseñanza autónoma.

- Ingeniería de software: Se centra en la creación de software que cumpla con los requisitos del cliente y se adapte a las necesidades del entorno.

Estas filosofías reflejan la diversidad de enfoques que los desarrolladores utilizan para abordar los desafíos del desarrollo de software, adaptándose a las necesidades del mercado y a las características del entorno de trabajo actual.